Ten Questions

What is the main goal of your work?	Providing a means of understanding concurrent program execution and structure combining the benefits of static concurrency analysis (SCA) with those of traditional trace-based debugging.
What are the tangible benefits to society from achieving the goal? (I.e. why should anyone pay for this?)	Concurrent programming is gaining in popularity as a way to exploit the power of multiprocessor machines that are becoming prevalent. Debugging these programs when they fail has proven to be difficult and expensive. Tools which aid in this will save much programmer effort and expense.
What are the technical problems that make the goal difficult to achieve? (ie. why hasn't this been done already?)	It requires the integration of techniques from two areas that have traditionally been viewed as exclusive choices: SCA vs. traditional debugging. SCA is a form of partial proof of correctness, and proponents attempt to expand these techniques for wider applicability, desiring to keep these formal properties intact. Traditional debugging proponents attempt to expand the familiar sequential debugging paradigm to concurrent programs. They note that complete SCA will always be intractable, and focus instead on simpler techniques with much wider applicability, though they give little formal insight into program structure.
What are the main elements of your approach?	Use of the Task Interaction Concurrency Graph as a central model to describe both individual executions and the overall structure of a concurrent program. This model is examined by the user, and extended under user control using both static analysis and dynamic tracing.
How does your approach handle the technical problems that have prevented progress in the past? (ie. what makes you think you can do it when no one else could before?)	The UCI Arcadia project has given us a well developed SCA platform to build from. This knowledge of SCA, combined with an extensive study of existing dynamic techniques gives me leverage to find the synergies between these greatly differing paradigms. I have decided to forego the partial proof properties offered by complete SCA to explore how incomplete information can be used to understand the behavior of a portion of the execution state space.
What are the unique, novel, and/or critical technologies developed in your approach?	The use of the TICG as a visible model for the programmer, the extensions to deal with partial TICGs, as well as the handling of dynamic features.
What are the potential spinoffs or other applications of your work?	The framework I develop could be combined with testing techniques. It could be used to enable the type of structural testing described by Taylor, Kelly and Levine. In combination with control of the runtime scheduler, you could actually have the program follow certain paths, and examine the possible behaviors dynamically.
How can progress be measured? (ie. how can anyone tell if/when you've succeeded?)	If the tools developed enable a debugging style based on the TICG model, and this model provides insights not available using existing techniques, then this work will have succeeded.
What have you accomplished thus far?	I have an initial prototype running that allows for trace collection and incremental expansion and examination of the partial TICG under user control.
What is your schedule for the work remaining.	I need to finish up the prototype to include dynamic tasking features, and debugging of data related problems. This should be completed in February. Demonstrating the techniques on larger problems and debugging the resulting problems will take another month. Finally I need to write a conference paper and a dissertation describing my work. This will take us into April, 1996.